Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Anej Lekše

Avtomatizacija delavniškega dnevnika

diplomsko delo

visokošolski strokovni študijski program
prve stopnje
računalništvo in informatika

Mentor: doc. dr. Andrej Brodnik

Ljubljana, 2020

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Preveri ali so glasovni asistenti v trenutnem stanju primerni za pomoč pri pisanju delavniških dnevnikov. Seznani se z obstoječimi programskimi rešitvami in jih analiziraj. Po analizi trga se loti izdelave svojega sistema za pisanje delavniških dnevnikov, ki vključuje glasovnega asistenta.

Na tem mestu zapišite, komu se zahvaljujete za izdelavo diplomske naloge. Pazite, da ne boste koga pozabili. Utegnil vam bo zameriti. Temu se da izogniti tako, da celotno zahvalo izpustite.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
API	Application Programming Interface	vmesnik za programiranje
AWS	Amazon Web Services	Amazonove spletne storitve
LIMS	Laboratory Information Management System	laboratorijski sistem za upravljanje informacij
MVVM	Model View View-Model	Model Pogled Pogled-Model
SNS	Simple Notification Service	preprosta storitev za opozorila
SQS	Simple Queue Service	preprosta vrstna storitev
UI	User Interface	uporabniški vmesnik
VUI	Voice User Interface	glasovni uporabniški vmesnik
XAML	Extensible Application Markup Langugage	razširljiv aplikacijski označitveni? jezik
XAML	Extensible Application Markup Langugage	razširljiv aplikacijski označitveni? jezik

Povzetek

Naslov: Avtomatizacija delavniškega dnevnika

Avtor: Anej Lekše

Diplomsko delo obravnava področje izboljšanja procesa pisanja delavniškega dnevnika ali laboratorijskega poročila. Največ časa anketirani študentje porabijo za prepisovanje v digitalno obliko, zapisovanje zapiskov na papir in urejanje teh zapiskov. Cilj je preizkusiti računalniški sistem z glasovnim asistentom, s pomočjo katerega lahko narekujemo zapiske med delom. Te zapiske pa lahko naknandno urejamo s pomočjo mobilne ali namizne aplikacije. Uporabili bomo Amazon Alexo zaradi enostavne izdelave lastnih programov (t.i. Skill-ov).

Ključne besede: mobilni razvoj, glasovni asistenti, razpoznava glasu, informacijski sistemi.

Abstract

Title: Workshop report automatisation

Author: Anej Lekše

This thesis deals with the process of optimising the process of writing a lab report. Students, that took part in the survey, spend the most time to type the report into a digital format, write notes on paper and ordering their notes. The goal of thesis is testing a system with a voice assistant that could be used to take notes during work itself. These notes can be edited and ordered later via a mobile or desktop application. We will use Amazon Alexa as it offers simple programming with Alexa Skills.

Keywords: mobile development, voice assistants, voice recognition, information systems.

Poglavje 1

Uvod

1.1 Opis domene raziskave

Diplomsko delo obravnava področje pisanja delavniških in laboratorijskih poročil. Delavniški dnevnik je dokument, ki opisuje potek izdelave izdelka po korakih. Zapis koraka dela vsebuje opis dela, uporabljena orodja in metode ter trajanje. Delavniški dnevnik lahko opisuje tudi korake kontrolnega postopka za željen izdelek.

V diplomski nalogi smo želeli izdelati in raziskati učinkovitost specializiranega sistema za pisanje delavniških dnevnikov. Ta sistem, bi sestavljali glasovni asistent, ki bi služil za narekovanje opomb, mobilne aplikacije, preko katere bi lahko urejali zapiske in strežnika.

// slika primer delavniškega dnevnik

1.2 Struktura diplomske naloge

// dodelaj na koncu

Diplomsko delo pričenjamo s predstavitvijo področja raziskave in kratko opišemo problem in možno rešitev. V sklopu te diplomski naloge bomo raziskali, ali so glasovni asistent in mobilna aplikacija primerno orodje za učinkovitejše pisanje delavniških dnevnikov. Začnemo z raziskavo že ob-

stoječih rešitve za to problematiko. Nato opišemo, kaj trenutne rešitve te problematike ponujajo in njihove prednosti in slabosti. Po analizi se lotimo opisa tehnologij, ki smo jih pri pisanju diplome uporabili. V naslednjem poglavju se lotimo načrtovanja sistema za pomoč pri pisanju laboratorijskih poročil. Natančno definiramo funkcionalnosti sistema, utemeljimo odločitev za izbiro Amazon Alexe in izdelamo Alexa Skill, API in mobilno aplikacijo. Funkcionalnosti sistema OpenReport testiramo in analiziramo. V predzadnjem poglavju opišemo možnosti nadaljnjega razvoja projekta.

Poglavje 2

Pregled problema in rešitve

2.1 Problem

2.1.1 Kaj je delavniški dnevnik?

// dodelaj na koncu

Opis tehnološkega postopka ali delavniški dnevnik je dokument, ki po korakih nedvoumno predstavi postopek izdelave izdelka. Delavniški dnevniki imajo lahko definirane tudi kontrolne postopke za izdelek, orodje, ki ga rabimo za izdelavo in seznam možnih nevarnosti pri delu.

Delavniški dnevnik je sestavljen iz zaporedja korakov, ki si sledijo v časovnem zaporedju. Vsak korak je sestavljata opis postopka in predvideno trajanje. Koraki lahko vsebujejo tudi slikovne razlage.

Opisi tehnološkega postopka se najpogosteje uporabljajo v proizvodnih obratih tovarn.

2.1.2 Primer

Spodnja slika prikazuje primer opisa tehnološkega postopka za izdelavo hladilnika.

// maskiran delavniški dnevnik iz gorenja

Ta opis tehnološkega postopka je strukturiran tako, da se vsi koraki sklicujejo na eno samo tehnično sliko.

// kako podrobno se opisuje korake?

2.1.3 Definicija problematike

V tej diplomski nalogi želimo raziskati trenutne metode izdelave opisa tehnološkega postopka. Raziskati želimo prednosti in slabosti trenutnih metod za izdelavo teh dokumentov. V kasnejših fazah raziskave želimo izdelati svoj specializiran sistem za pisanje delavniških dnevnikov, ki bo poskušal popraviti slabosti trenutnih metod.

Izhajajoč iz navedenega opredeljujem problem diplomskega dela: kako lahko proces izdelave opisa tehnološkega postopka naredimo učinkovitejši in prijaznejši uporabniku s pomočjo specializiranega sistema?

2.2 Analiza obstoječih rešitev

2.2.1 Papir in pisalo

Najstarejša metoda za izdelavo takšnega dokumenta je zapis na formuliran list papirja (// citiraj sliko).

// slika delavniškega dnevnika na papirju

Prednosti uporabe papirja in pisala so, da pri delu izdelovalec ne potrebuje računalnika in cenovna ugodnost. Slabosti takšnega postopka so:

- omejitve glede velikosti prostora, namenjenega vsakemu koraku,
- problematično dopisovanje in urejanje obstoječih korakov,
- nečitljivost rokopisa,
- občutljivost papirja na fizične poškodbe (trganje, mečkanje, vnetljivost,...).

// testni primer za bolt mod na papir

2.2.2 Pisarniški programi

Opis tehnološkega postopka lahko izdelamo v pisarniških programih kot so Microsoft Word ali LibreOffice Writer.

Ta pristop reši večino slabosti uporabe papirja in pisala za pisanje opisa tehnološkega postopka. Korake lahko enostavno dodajamo in urejamo. Prav tako je možno dodajati slikovno gradivo. Pisarniški programi omogočajo tudi enostaven izvoz dokumenta na tiskalnik, če želimo imeti dokument na listu papirja.

Kljub temu pa uporaba te metode prinese nove slabosti:

- če imamo dokument na več mestih, moramo ob spremembah zagotoviti, da se spremenijo vsi.
- Slikovno gradivo je vezano na dokument. Ob spremembah moramo spremeniti celoten dokument, ne le slike.
- Možnost izgube ali izbrisa podatkov.

// slika bolt mod v writerju

2.2.3 Specializirani moduli za poslovne informacijske sisteme

Podjetja in tovarne za svoje izdelke večinoma uporabljajo specializirane module, tesno povezavne z njihovimi informacijskimi sistemi.

// maskirana slika gorenje

Primer je prikazan modul za informacijski sistem SAP (// citiraj sap). Do tega opisa tehnološkega postopka se lahko dostopa iz računalnikov na delovnih mestih, kjer se izvajajo koraki, opisani v dokumentu. Opis tehnološkega postopka v tem primeru sestavljajo:

- podatki o izdelku,
- dodatne opombe,

- opisi korakov,
- definicija kontrolnih postopkov in pregleda,
- CAD izris izdelka.

Takšen specializiran sistem je tesno povezan s proizvodno linijo, prilagodljiv in optimiziran s stališča funkcionalnosti. V takšnih sistemih se podatki hranijo na strežniku kar pomeni, da uporabniki sistema nimajo težav z razkropljenostjo podatkov. Glavne slabosti takšnega sistema so:

- potrebna proizvodna infrastruktura, ki jo sistem rabi za optimalen izkoristek in
- cena, ki je potrebna za implementacijo.

V kemijski industriji se laboratorijska poročila pišejo s pomočjo LIMS (Laboratory Information Management System). Sistemi kot so OpenLIMS (// citiraj) imajo že vključene module za pisanje poročil (// citiraj)

// testiraj LIMS

Poglavje 3

Načrtovanje in razvoj sistema za pisanje opisov tehnoloških postopkov

3.1 Definicija funkcionalnosti

Učinkovit sistem za pisanje opisa tehnološkega postopka:

- povzroča čim krajše prekinitve dela,
- vsebuje možnost uporabe glasu za narekovanje korakov,
- hrani uporabljene podatke (fotografije, zapiske) na enem mestu.

Sistem, ki ga bom sprogramiral v tej raziskavi bo namenjen predvsem individualnim uporabnikom in bo mišljen kot alternativa pisanju delavniških dnevnikov na list papirja ali s pisarniškimi programi.

Podpirati mora ustvarjanje novega dnevnika in odpiranje ter urejanje že ustvarjenih dnevnikov.

Vsak opis tehnološkega postopka mora imeti naslov, opis, seznam možnih nevarnosti pri delu in seznam korakov dela. Seznam korakov dela mora podpirati dodajanje novih korakov, urejanje obstoječih korakov, brisanje obstoječih korakov in spreminjanje vrstnega reda korakov.

Da se čas pri delu bolje izkoristi, bo korake za trenutno odprt delavniški dnevnik možno dodajati tudi preko glasovnega asistenta. Poleg narekovanja besedilnih korakov bo preko glasovnega asistenta možno tudi odpreti kamero ali obrazec za dodajanje koraka.

Raziskava iz leta 2018 je pokazala izboljšano učinkovitost pri delu raziskovalcev v kemijskem laboratoriju, v katerega so integrirali glasovne pomočnike Namen raziskave je bil preizkus praktične uporabnosti glasovnih asistentov za naloge, kot so branje laboratorijskih postopkov po korakih, in glasovno upravljanje laboratorijskih instrumentov. Pozitivni rezultati bi lahko bili ključnega pomena za slabovidne člane laboratorijev. Kot glasovni asistent je bila uporabljena Amazon Alexa. Prepoznavanje govora in ukazov je bilo konsistentno in hitro, ne glede na spol uporabnika. Motnje pri razpoznavanju je povzročal večinoma ozadni hrup. Povprečna natančnost prepoznave ukazov je bila 95%. Raziskovalci so zabeležili tudi problem moteče kakofonije v laboratoriju, v katerem je več raziskovalcev, ki uporabljajo glasoven nadzor naprav.

3.2 Načrt

Sistem, za pisanje opisov tehnoloških postopkov sem poimenoval OpenReport. Sistem OpenReport bodo sestavljali:

- strežniški program,
- mobilna aplikacija,
- glasovni asistent.

Strežnik bo v podatkovni bazi hranil uporabnike, delavniške dnevnike in korake. Ponujal bo REST API, ki ga bosta uporabljala glasovni asistent in mobilna aplikacija za operacije nad delavniškimi dnevniki.

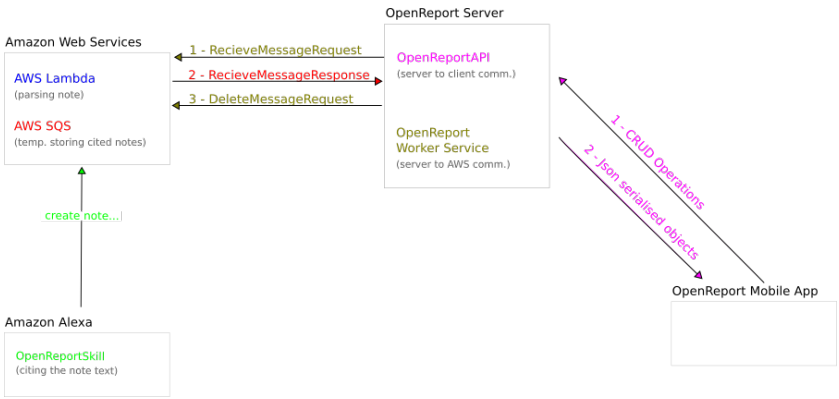
Preko mobilne aplikacije bo uporabnik lahko:

- opravil regsitracijo in prijavo,

- ustvaril nov delavniški dnevnik,
- odprl obstoječe delavniške dnevnike,
- ustvaril in urejal korake delavniškega dnevnika,
- zajemal slike in jih vstavljal v delavniški dnevnik,
- brisal korake delavniškega dnevnika,
- urejal vrstni red korakov delavniškega dnevnika.

Za glasovni asistent Amazon Alexa bomo razvili "Skill", s katerim bo uporabnik lahko:

- v odprto poročilo vstavil dobesedno narekovan korak,
- odprl obrazec za dodajanje novega tekstovnega koraka,
- odprl kamero in obrazec za dodajanje koraka s fotografijo.



Slika 3.1: Visokonivojski načrt sistema

3.3 Uporabljene tehnologije in programska oprema

3.3.1 .NET

.NET je razvojna platforma, razvita s strani Microsofta. Obsega programske jezike, prevajalnike, orodja in knjižnice, ki omogočajo širok spekter primerov uporabnosti, hkrati pa se ohranja enovitost ozadne kode.

Tehnologije .NET ogrodja, ki sem jih uporabil v tej diplomski nalogi so:

- .NET Core - odprtokodna platforma za razvoj spletnih storitev,
- Xamarin, ki je ogrodje za razvoj mobilnih aplikacij za najpogostejše mobilne operacijske sisteme (Android, iOS).

.NET sem izbral, saj je zelo dobro integriran z Amazonovim AWS API-jem in ker je dobro dokumentiran.

3.3.2 Xamarin Forms

Ogrodje Xamarin je odprtokodno orodje za razvoj mobilnih aplikacij, ki ga je razvil Microsoft.

Z ogrodjem Xamarin je mogoče pri deliti večino ozadne in ospredne kode med različnimi mobilnimi operacijskimi sistemi. Za ozadno kodo se uporablja .NET (C#), za front-end pa se uporablja XAML (Extensible Application Markup Language).

Xamarin sem izbral, saj imam z njim že predhodne izkušnje.

// še kakšen razlog?

3.3.3 Amazon Alexa

Amazon Alexa je glasovni asistent, razvit s strani podjetja Amazon. Za Amazon Alexo sem se odločil, saj ponuja enostavno možnost programiranja s "Skill-i". Poleg tega je Alexo enostavno integrirati z drugimi Amazonovimi spletnimi storitvami, kot so AWS SQS.

3.3.4 Alexa Skill

Alexine osnovne funkcionalnosti lahko nadgradimo s programi, ki se jim reče "Skill". Da lahko ustvarimo in objavimo Skill rabimo račun Amazon razvijalca (*ang. Amazon Developer Account*).

"Skill" sestavljajo:

- Invocation - fraza, ki "Skill" zažene,
- Intent - fraze, ki jih Skill razpozna kot funkcije,
- Endpoint - omrežni vir, kjer se nahaja ozadna koda "Skill-a".

Ko Alexa zasliši Invocation ali katerega od Intentov, glasovni posnetek pošlje na Amazonov strežnik. Ta s pomočjo glasovnega razpoznavnega modela prepozna ukaze in pošlje poseben zahtevek na Endpoint. To je lahko druga Amazonova storitev (npr. AWS Lambda), storitev na Microsoftovem Azure strežniku ali naš lasten strežnik, dostopen preko javne domene.

Ko Endpoint obdela zahtevo, se odgovor pošlje nazaj na Amazonov strežnik v obliki znakovnega niza. Ta podatek se nato pošlje nazaj na uporabnikovo Alexo, ki prejeti znakovni niz "izgovori".

3.3.5 Amazon Web Services

AWS je skupek oblačnih storitev, ki ga ponuja podjetje Amazon. Ponuja integracijo s popularnimi programskimi jeziki in ogrodji, kot so Java, .NET, Python in Node.js preko AWS API storitve. Pri tej diplomski nalogi sem se osredotočil na dva sistema iz skupka AWS.

AWS SQS

AWS SQS je sistem za pošiljanje tekstovnih sporočil med odjemalci preko Amazonovih strežnikov. Za hranjenje sporočil je treba registrirati SQS Queue *(slo. SQS vrsto)*. Ta je lahko neurejena vrsta, kjer prejeta sporočila niso

nujno urejena po času ustvarjanja, lahko pa je tipa FIFO, pri kateri je zagotovljeno, da dobimo vsa sporočila v vrstnem redu, v katerem so bila poslana. V sklopu te diplome sem uporabil vrsto FIFO.

To storitev bom uporabil za komunikacijo med Amazon Alexo in OpenReport strežnikom.

// ali še opišem SQS?

AWS Lambda

Ozadno kodo za Alexa Skill sem gostil na platformi AWS Lambda. To je storitev za gostovanje dogodkovno vodene ozadne kode. Za to platformo smo se odločili zaradi dobre integracije z Alexa Skill Kit-om in razvojnim orodjem Visual Studio.

// ali še opišem Lambdo?

3.4 Strežnik

Za uporabo centralnega strežnika smo se odločili, da lahko do hranjenih podatkov dostopamo iz različnih naprav preko enotnega vmesnika (API). V našem primeru bosta s strežnikom komunicirala mobilna aplikacija in glasovni asistent. Podatke bomo hranili v podatkovni bazi, ki jo streže SQL Server.

Baza hrani tabele Users (*slo. Uporabniki*), Projects (*slo. Projekti*) in Notes (*slo. Zapiski*). Vsak uporabnik lahko ima 0 ali mnogo projektov (opisov tehnološkega postopka). Vsak projekt ima lahko 0 ali mnogo zapiskov (korakov).

// naredi sliko sheme baze

Strežnik ima naslednje naloge:

1. komunikacija s podatkovno bazo,
2. komunikacija z odjemalcem (mobilno aplikacijo),
3. komunikacija z glasovnim asistentom (Amazon Alexa),

4. ponujanje vmesnika, preko katerega lahko odjemalci delajo s podatki v bazi.

3.4.1 Implementacija API

Komunikacija med strežnikom in odjemalci poteka preko HTTP po pristopu API. Podatki se prenašajo v formatu JSON. To metodo smo izbrali zaradi enostavnosti implementacije in možnosti širjenja v prihodnosti.

Glede na to, da lahko sistem uporablja več uporabnikov, sem se odločil implementirati sistema za avtentikacijo uporabnikov in avtorizacijo zahtev. Avtentikacija z uporabniškim imenom in geslom omogoča preverjanje identitete. Ko se vzpostavi zaupanje, se prijavljenemu uporabniku dodeli avtorizacijski žeton, iz katerega se da razbrati, za katerega uporabnika gre in katere pravice so mu dodeljene.

Registracija in prijava

Odjemalec mora poslati na API objekt razreda *RegisterUserRequest*. Odjemalec ga mora poslati na URL "`/identity/register`".

```
RegisterUserRequest {
string Email;
string Password;
}
```

Ko strežnik prejme ta objekt, ga pošlje v avtentikacijsko storitev. V tej storitvi preveri, ali uporabnik že obstaja. Če obstaja, se zabeleži napaka in nadaljnja registracija prekine. Če ta uporabnik ne obstaja, se polje `Password` zakriptira in se skupaj s poljem `Email` zapiše v bazo. Uporabnik pa pri registraciji dobi tudi svoj unikaten identifikator `UserID`.

V kolikor se je v avtentikacijski storitvi dogodila kakršna koli napaka, se klientu vrne objekt razreda `AuthFailedResponse`

```
AuthFailedResponse {
IEnumerable<string> Errors;
}
```

V tem objektu se odjemalcu v zbirki pošljejo vse napake, ki jih je avtentikacijska storitev zabeležila pri neuspešni registraciji. Če je bila registracija uspešna, se odjemalcu pošlje objekt razreda `AuthSuccessResponse`

```
AuthSuccessResponse {
string UserId;
string Token;
}
```

V polje `Token` se zapiše avtorizacijski žeton. Žeton je tipa JWT ali *JSON Web Token*. Sestavljajo ga e-mail uporabnika, uporabnikov unikatni identifikator `UserID`, čas zapada žetona in tip simetričnega kodiranja, uporabljenega za enkripcijo žetona.

Ta objekt se nato pošlje nazaj odjemalcu. Odjemalec nato ta žeton doda glavi vseh svojih HTTP zahtevkov na OpenReport strežnik. V nadaljnji komunikaciji, strežnik iz žetona razbere uporabnikov unikatni identifikator in ga uporabi pri poizvedbah po podatkovni bazi.

Prijava poteka podobno. Uporabnik mora poslati na API objekt *LoginUserRequest*. URL, na katerega mora odjemalec poslati ta objekt, je `/identity/login`.

```
LoginUserRequest {
string Email;
string Password;
}
```

Ko strežnik prejme ta objekt, ga pošlje v avtentikacijsko storitev. V tej storitvi preveri, ali uporabnik že obstaja. Če ta uporabnik ne obstaja, ali pa je njegovo zakriptirano geslo v podatkovni bazi drugačno kot to, kar je v polju `Password`, se zabeležijo napake in prijava se prekine. Odjemalec prejme objekt razreda `AuthFailedResponse`.

Če uporabnik obstaja in se njegovo zakriptirano geslo iz polja `Password` ujema z geslom v podatkovni bazi, je avtentikacija uspešna. Odjemalec prejme objekt razreda `AuthFailedResponse`.

V primeru uspešne avtentikacije se avtorizacijski žeton `Token` generira in

pošlje enako kot pri registraciji.

Operacije z delavniškimi dnevniki

Vsak uporabnik lahko ima nič ali več delavniških dnevnikov.

Vse operacije nad uporabnikovim delavniškim dnevnikom morajo biti avtorizirane. V kolikor niso, bo API vedno javil napako `Bad Request: Not Authorised`.

Pri ustvarjanju novega opisa tehnološkega postopka (v nadaljevanju poglavja *projekta*) mora uporabnik podati naslov, kratek opis projekta in seznam možnih nevarnosti.

To odjemalec zapiše v objekt razreda `CreateProjectRequest`.

```
CreateProjectRequest {
string Title;
string Description;
string Dangers;
}
```

Ta objekt se nato pošlje preko POST metode na strežnik na naslov `/projects/create`.
Storitev za operacije nad projekti nato kreira nov objekt razreda `Project` s podatki iz prejete zahteve. V primeru, da je zahteva ustrezno formulirana in avtorizirana, se projekt zapiše v podatkovno bazo. Odjemalcu se kot odgovor pošlje kopija tega projekta.
Razred `Project` izgleda tako:

```
Project {
int Id; // unikatni identifikator
string Title;
string Description;
string Dangers;
IEnumerable<Note> Notes; // seznam korakov
...
}
```

Do kateregakoli svojega projekta lahko odjemalec v nadaljnji komunikaciji

dostopa tako, da pošlje GET zahtevek na URL `/projects/{id}`. Polje `id` mora v tem primeru biti unikatni identifikator projekta.

Za izbris projekta mora uporabnik poslati DELETE zahtevo na URL `/projects/{id}`. Če zahteva ni avtorizirana z ustreznim žetonom, se projekt ne izbriše.

Operacije nad koraki opisa tehnološkega postopka

Vsak projekt ima lahko nič ali več korakov. Vsak korak ima naslov, opis in predvideno trajanje.

Korak je lahko izključno tekstovni, lahko pa ima tudi pripadajočo sliko.

Da ustvarimo tekstovni korak moramo poslati objekt razreda `Note` preko POST metode na URL `/projects/{id}/addnote`. Polje `id` mora biti unikatni identifikator projekta, katermu želimo dodati korak.
Razred `Note` izgleda tako.

```
Note {
...
string Title;
string Text;
int Hours;
int Minutes;
int Seconds;
...
}
```

Strežnik najprej preveri, če je pošiljatelj poslal avtorizirano zahtevo. Iz avtorizacijskega žetona strežnik razbere uporabnikov `UserID`. Če je projekt z identifikatorjem `id` res v lasti uporabnika z identifikatorjem `UserID`, se bo korak dodal v zbirko korakov tega projekta.

Če želimo ustvariti slikovni korak, moramo poslati objekt razreda *AddImageRequest* na URL `/projects/{id}/addimage`.

```
AddImageRequest {
Note note;
```

```
string ImageString;
}
```

V tem razredu predstavlja polje `ImageString` pripadajočo fotografijo, zakodirano v znakovni niz. Za kodirni algoritem sem uporabil Base64.

Za hranjenje tekstovnih in slikovnih korakov sem zaradi preprostosti implementacije uporabil isti razred `Note`. Tekstovni in slikovni korak se ločita v vrednosti boolean zastavice `IsImage`. Tekstovni korak ima to zastavico nastavljeno na vrednost `false`, slikovni pa `true`.

```
Note {
bool IsImage;
string Title;
string Text;
int Hours;
int Minutes;
int Seconds;
string Url; // Lokacija slike na klientu
string ServerUrl; // Lokacija slike na strežniku
...
}
```

Poljema `Url` in `ServerUrl` se dodeli vrednost samo pri slikovnih korakih. Ko se na klientu zajame fotografija, klient nastavi vrednost polja `Url` na lokacijo ustvarjene fotografije v datotečnem sistemu.

Polje `ServerUrl` se nastavi šele na strežniku. Ko strežnik prejme zahtevo za kreiranje slikovnega koraka, se korak `Note` prebere iz objekta `AddImageRequest` in zapiše v podatkovno bazo. Nato se `ImageString` dekodira in hrani na strežniku. Ko se fotografija uspešno zapiše v datotečni sistem, se v polje `ServerUrl` zapiše lokacija pravkar ustvarjene datoteke na strežniku in spremembe se hranijo v podatkovni bazi.

Urejanje in brisanje in korakov v projektu

Da se korak v projektu izbriše, moramo poslati avtorizirano zahtevo na URL `/projects/{pid}/delete/{nid}`. V tem primeru je polje `pid` unikatni identifikator projekta, v katerem se nahaja korak, `nid` pa unikatni identifikator objekta `Note`, ki ga želimo izbrisati. V kolikor ima ta objekt `Note` postavljeno zastavico `IsImage` na `true`, se poleg zapisa v bazi izbriše tudi pripadajoča slikovna datoteka.

Pri posodabljanju (urejanju) korakov, moramo poslati avtorizirano zahtevo na URL `/projects/{pid}/update/{nid}`. Telo zahteve mora vsebovati objekt razreda `Note`.

// heyhey kaj pa slika?

Spreminjanje vrstnega reda korakov v projektu

Položaj koraka `Note` v projektu lahko razberemo iz atributa `Position`. Prvi korak ima `Position` 0, drugi 1, itd.

```
Note {
int Id;
int Position;
bool IsImage;
string Title;
string Text;
int Hours;
int Minutes;
int Seconds;
string Url;
string ServerUrl;
}
```

Pri dodajanju korakov v projekt se obstoječe korake projekta razvrsti po vrednosti polja `Position`. Vzamemo največjo vrednost tega polja in ji prištejemo 1. Nato to vrednost priredimo polju `Position` novo kreiranega koraka.

Ko želimo spremeniti pozicijo koraka v projektu, moramo poslati zahtevo PUT na URL `/projects/{pid}/{nid}/{positions}`.

V tem primeru je polje `pid` unikatni identifikator projekta, v katerem se nahaja korak, `nid` unikatni identifikator koraka `Note`, in `Positions` število mest za kolikor ga želimo prestaviti. To število je lahko pozitivno ali negativno celo število. Negativna vrednost prestavi korak proti začetku seznama, pozitivna pa proti koncu.

Izvoz projektov

Projekt lahko izvozimo v dva formata, v tekstovno datoteko in HTML dokument.

To naredimo tako, da pošljemo avtorizirani GET zahtevi na URL-ja `/projects/{id}/export/text` ali `/projects/{id}/export/html`.

Pri izvozu v tekstovno datoteko storitev za upravljanje projektov na strežniku ustvari novo tekstovno datoteko. Najprej vanjo vpiše naslov in opis projekta ter možne nevarnosti pri delu. Nato uredi korake po vrednosti stolpca `Position` in enega za drugim zapiše v datoteko. Če je korak slikovni, se zapiše tudi lokacija pripadajoče fotografije.

Pri izvozu v HTML dokument se naslov zapiše kot HTML naslov H1, opis kot naslov H2, in koraki kot HTML odstavki. V HTML dokumentu lahko prikažemo poleg slikovnih korakov tudi slike same.

// slika exportanega

Lokacija izvožene datoteke se nahaja v polju `FolderLocation` v razredu `Project`. Privzeta lokacija, ki se nastavi ob kreiranju projekta je "`C:/users/USER/Public Documents/OpenReport`".

```
Project {
int Id;
string Title;
string Description;
string Dangers;
string FolderLocation; // lokacija fotografij in izvoženih datotek
```

```
IEnumerable<Note> Notes;
...
}
```

3.4.2 Komunikacija strežnika z glasovnim asistentom

Strežnik z glasovnim asistentom Amazon Alexa komunicira preko storitve AWS SQS. Glasovni asistent od uporabnika prejme glasovni ukaz. Na podlagi tega ukaza lahko glasovni asistent na SQS vrsto odloži tri različna tekstovna sporočila:

- "addnote",
- "addimage",
- "{narekovano besedilo}",

// ali tukaj opišem kaj delajo??

Utemeljitev uporabe SQS

Za hranjenje sporočil v storitvi SQS smo se odločili po primerjavi z dvema alternativnima pristopoma. Prvi je bil odpiranje omrežne vtičnice (ang. Web Socket). Ta bi prinesla učinkovitejši prenos podatkov, implementacija pa bi bila težja. Druga alternativa pa je bila direktna komunikacija Amazon Alexe in našega strežnika preko REST API zahtevkov, ki se prenašajo po protokolu HTTP. Ta alternativa bi bila implementacijsko preprostejša kot uporaba SQS. Problem te alternative bi bil, da bi bilo treba javno odpreti API dostopne točke, ki jih lahko asistent uporablja, ali implementirati nove metode za avtentikacijo in avtorizacijo zahtevkov iz glasovnega asistenta.

Zahtevanje in izpust glasovnega asistenta

V sklopu našega projekta bomo na enem OpenReport strežniku uporabljali samo enega glasovnega asistenta. Tega asistenta lahko uporabnik "zahteva"

za lastno uporabo in ga po uporabi "izpusti". Ali ima določen uporabnik nase vezanega asistenta, vidimo po vrednosti zastavice `UsesVoiceAssistant` v njegovem objektu razreda `User`.

Ko uporabnik odpre katerega od svojih projektov, se unikatni identifikator tega projekta nastavi v polje `LastOpenedProjectId` v njegovem objektu razreda `User`.

Zahtevki, ki jih strežnik prejme od glasovnega asistenta, se navezujejo na ta projekt.

```
User :  IdentityUser {
string Id;
string Email;
string Password;
IEnumerable<Project> Projects;
...
int LastUsedProjectId;
bool UsesVoiceAssistant;
}
```

Delovanje

Na strežniku OpenReport teče storitev za komunikacijo z glasovnim asistentom. Ta storitev vsakih deset sekund na SQS pošlje zahtevek `RecieveMessageRequest`.

```
RecieveMessageRequest {
AttributeName
MaxNumberOfMessages
QueueUrl
WaitTimeSeconds
...
}
```

AWS SQS kot odgovor vrne objekt tipa `RecieveMessageResponse`.

```
RecieveMessageResponse {
IEnumerable<Message> Messages;
```

```
...
}
```

V tem odgovoru se nahaja seznam sporočil, ki čakajo na sprejem iz SQS vrste. V kolikor je v odgovoru vsaj eno sporočilo, pregledamo telo vseh sporočil.

Ko sporočila sprejmemo, jih moramo izbrisati iz SQS vrste. Če jih sproti ne izbrišemo, iz FIFO vrste ne moremo brati najnovejših sporočil, ampak le 5 najstarejših.

Sporočila izbrišemo iz vrste preko zahtevka `DeleteMessageRequest`. AWS SQS storitev v odgovor vrne `DeleteMessageResponse`, a v tej diplomski nalogi tega odgovora ne uporabljamo naprej.

Zahtevki, ki jih prejmemo od glasovnega asistenta v projektu, ki ga je uporabnik z glasovnim asistentom nazadnje odprl, nastavijo vrednosti treh boolean zastavic v objektu razreda `Project`.

- Zahtevek "`addnote`" nastavi zastavico "`NoteRequest`" na `true`,
- zahtevek "`addimage`" nastavi zastavico "`ImageRequest`" na `true`,
- zahtevek "{`narekovano besedilo`}" nastavi zastavico "`AlexaNoteRequest`" na `true` in prejeto besedilo doda projektu kot korak.

```
Project {
int Id;
string Title;
string Description;
string Dangers;
string FolderLocation;
IEnumerable<Note> Notes;
bool NoteRequest;
bool ImageRequest;
bool AlexaNoteRequest;
}
```

Če je vsebina prejetega sporočila enaka `addnote`, storitev preveri, kateri uporabnik ima trenutno nase vezanega glasovnega asistenta. Nato v projektu, katerega je nazadnje odprl nastavi vrednost zastavice `NoteRequest` na `true`.

Če je vsebina enaka "`addimage`", storitev preveri, kateri uporabnik ima trenutno nase vezanega glasovnega asistenta. Nato v projektu, katerega je nazadnje odprl nastavi vrednost zastavice `ImageRequest` na `true`.

Če vsebina prejetega sporočila ni enaka "`addimage`" ali "`addnote`", pomeni, da je prejeto sporočilo dobesedno narekovan korak. Kot prej, storitev preveri, kateri uporabnik ima nase vezanega glasovnega asistenta. Nato v projektu, katerega je nazadnje odprl nastavi vrednost zastavice `AlexaNoteRequest` na `true`. Poleg tega ustvari nov objekt razreda `Note`, ki ima naslov "Voice note", vsebina pa je telo prejetega sporočila. Nov objekt se nato vstavi v najden projekt.

3.5 Implementacija „Alexa Skilla"

// narekovaje bom popravil, ko bo vsebinsko ok

Alexa Skill bomo uporabili za glasovno upravljanje aplikacije. Glavni namen Alexa Skilla v sklopu sistema OpenReport bo dodajanje korakov v opis tehnološkega postopka.
Natančneje, Skill bo uporabniku omogočal:

- dodajanje tekstovnega koraka,
- odpiranje kamere in dodajanje slikovnega koraka in
- dobesedno narekovanje besedila tekstovnega koraka.

Ko bo Alexa zaslišala ukaz za zagon skilla (ang. Invocation), se bo Skill začel izvajati. Ta fraza se glasi "*make a report note*".

Vsak nadaljnji ukaz, ki ga bo uporabnik izrekel, preden se Skill preneha izvajati, se bo primerjal s frazami za zagon podprogramov Skilla. Te fraze

se imenujejo "namere" (ang. Intent). V našem skillu bomo imeli tri glavne namere:

- `TakeNoteIntent`,
- `OpenTextNoteFormIntent`,
- `OpenImageNoteFormIntent`.

// slike intentov v alexa developer console

`TakeNoteIntent` se bo zagnal, ko bo po invokacijski frazi uporabnik izrekel "`take note {besedilo}`" ali "`note {besedilo}`". `{besedilo}` je tekstovna spremenljivka, v katero Skill hrani razpoznano tekstovno vsebino koraka. V primeru, da uporabnik izreče "*note unscrew the backplate*", bo vrednost spremenljivke `{besedilo}` "unscrew the backplate".

Ta Intent bo v ozadni kodi Skill-a zagnal funkcijo, ki bo ustvarila novo SQS sporočilo. V telo tega sporočila bo funkcija vstavila vrednost spremenljivke `{besedilo}` in ga poslala v SQS vrsto. Skill zatem vrne uporabniku odgovor "Noted!" in se preneha izvajati.

`OpenTextNoteFormIntent` se bo zagnal, ko bo po invokacijski frazi uporabnik izrekel "`create a text note`".

Ta Intent bo v ozadni kodi Skill-a zagnal funkcijo, ki bo ustvarila novo SQS sporočilo. V telo tega sporočila bo funkcija vstavila vrednost "`addnote`" in ga poslala v SQS vrsto. Skill zatem vrne uporabniku odgovor "Opening the form!" in se preneha izvajati.

`OpenImageNoteFormIntent` se bo zagnal, ko bo po invokacijski frazi uporabnik izrekel "`take a picture`".

Ta Intent bo v ozadni kodi Skill-a zagnal funkcijo, ki bo ustvarila novo SQS sporočilo. V telo tega sporočila bo funkcija vstavila vrednost "`addimage`" in ga poslala v SQS vrsto. Skill zatem vrne uporabniku odgovor "Launching camera!" in se preneha izvajati.

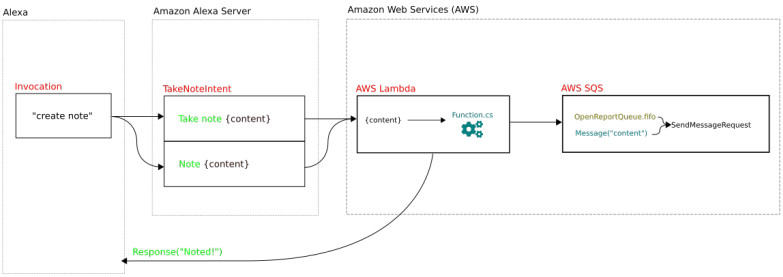
3.5.1 Izvajanje "Skill-a" po korakih

Alexa Skill se bo začel izvajati, ko uporabnik izreče definirano invokacijsko frazo. Alexa posnet govorni ukaz pošlje na Amazonov Alexa Server. Tam se s pomočjo NLU razpoznavnega modela poskusi pretvoriti v znakovni niz. Ta znakovni niz se primerja z vsemi definiranimi invokacijskimi frazami za Skill-e, ki so vezani na naš Amazon račun.

Če Alexa Server uporabnikov glasovni ukaz razpozna kot ,,make a report note", se začne izvajati OpenReportAlexaSkill. V ozadno kodo Skill-a (ang. Endpoint) se pošlje zahteva tipa `LaunchRequest`. Endpoint nastavimo v Alexa Developer Console. Lahko je AWS Lambda funkcija, lahko pa je naš lasten strežnik, dostopen preko javne domene. Endpoint je v našem primeru gostovan na storitvi AWS Lambda. Ob prejetju `LaunchRequest` Endpoint našega Skilla vrne vprašanje "What now?".

Naslednji glasovni ukaz se primerja s frazami za namere (ang. Intent). Če se ukaz ujema z frazami, ki zaženejo katerega od Intent-ov TakeNoteIntent, OpenTextNoteFormIntent ali OpenImageNoteFormIntent, se na Endpoint pošlje zahteva tipa `IntentRequest`.

Iz prejetega `IntentRequest`-a nato dobimo ime Intent-a. Na podlagi imena prejetega Intent-a ločimo, ali gre za TakeNoteIntent, OpenTextNoteFormIntent ali OpenImageNoteFormIntent in zaženemo funkcije, opisane v prejšnjem poglavju.



Slika 3.2: Načrt Alexa Skilla

3.6 Implementacija mobilne aplikacije

Mobilna aplikacija bo služila kot osnoven način za interakcijo s strežnikom. Do strežnika bo dostopala preko API vmesnika. Zahtevki se bodo prenašali preko HTTP protokola.

Mobilno aplikacijo sem se odločil razviti s tehnologijo Xamarin.Forms. Razlogi za to so predvsem dobro poznavanje tehnologije in enostavna integracija z ostalimi tehnologijami iz .NET sklopa.

3.6.1 Pristop

Pristop razvoja aplikacije, ki smo ga uporabili za programiranje aplikacije, se imenuje Model View View-Model (v nadaljevanju MVVM). Pri tem pristopu aplikacijo razdelimo na tri dele.

Model je del, kjer definiramo elemente naše "poslovne logike" (opis tehnološkega postopka, korak, uporabnik,...).

View je uporabniški vmesnik, ki ga vidi uporabnik.

ViewModel pa se uporablja, da se poveže funkcije uporabniškega vmesnika in modela ("poslovne logike") ter po potrebi preoblikuje podatke.

Rezultat upoštevanja tega pristopa je čista koda, ki se ne prepleta. Model vsebuje le abstrakcijo naših podatkov in poslovno logiko. Ti podatki se v ViewModelu pretvorijo v obliko, ki bo prikazana uporabniku. View pa vsebuje izključno grafične elemente.

// slika mvvm

3.6.2 Povezava na strežnik, prijava in registracija uporabnika

Ob zagonu mobilne aplikacije, uporabnika pričaka stran za povezavo na OpenReport strežnik. V tekstovno polje mora vpisati URL ali IP strežnika. Ta naslov bomo v nadaljevanju podpoglavja označevali z `naslovstrežnika`.

// slika povezavne strani

Ko uporabnik klikne gumb "Connect", se preveri, ali je naslov pravilnem formuliran. Če je, storitev za komunikacijo s strežnikom pošlje neavtorizirano zahtevo na "`naslovstrežnika`/connect". Če je na tem naslovu res OpenReport strežnik, se kot odgovor odjemalcu pošlje boolean vrednost `true`.

Storitev za komunikacijo s strežnikom si nato hrani vrednost `naslovstrežnika` kot predpono vseh nadaljnjih zahtevkov.

Ob neuspešnem poskusu povezave se uporabniku na strani prikaže opis napake. Če je poskus uspešen, se stran za povezavo na strežnik zapre.

Prikaže se stran za prijavo. Na tej strani uporabnik vpiše svoje uporabniško ime ali geslo, lahko pa odpre stran za registracijo računa.

Poskus prijave poteka tako, avtentikacijska storitev uporabniško ime in geslo iz prijavne strani zapiše v objekt razreda `LoginUserRequest`.

```
LoginUserRequest {
Email;
Password;
}
```

Nato se ta objekt pošlje na strežnik na URL `/identity/login` preko POST metode. Strežnik ob uspešni avtentikaciji vrne objekt razreda `AuthSuccessResponse`. Iz tega objekta uporabnik prebere svoj avtorizacijski žeton. Ta žeton hrani

avtentikacijska storitev. Pri nadaljnjih poizvedbah komunikacijska storitev ta žeton doda v glavo HTTP zahtevkov.

Če je avtentikacija neuspešna, strežnik vrne objekt razreda `AuthFailedResponse`, v katerem je podan opis napake. Ta opis napake se nato uporabniku izpiše na prijavni strani.

Stran za registracijo je podobna strani za prijavo, le da ima dve vnosni polji za vpis gesla. Drugo vnosno polje je preventivni ukrep, da se uporabnik ne registrira z napačno vtipkanim geslom.

Registracijski zahtevek se pošlje na strežnik na URL `/identity/register`. Zahtevek je objekt razreda `RegisterUserRequest`.

```
RegisterUserRequest {
Email;
Password;
}
```

// dodaj sliko registracijske strani in prijavne strani

3.6.3 Ustvarjanje, brisanje in odpiranje projektov

3.6.4 Zahtevanje in izpust glasovnega asistenta

3.6.5 Operacije znotraj projekta

3.6.6 Povezava na strežnik in razred Startup.cs

// razlozi kot normalen človek

Za povezavo na strežnik sem želel v projektu imeti eno instanco statičnega razreda. Ta razred (OpenReportCloudCommunicationService) vsebuje metode, namenjene komunikaciji z API-jem.

Registriral sem ga po metodi DependencyInjection. To sem dosegel tako, da sem v projekt dodal razred Startup.cs. V tem razredu sem registriral razred ServiceProvider, s katerim bom lahko dostopal do registriranih razredov drugod po aplikaciji.

V ServiceProvider sem dodal dva razreda. Prvi je HttpClient, ki sem ga uporabil za pošiljanje HTTP sporočil na strežnik, drugi pa je OpenReportServerCommunicationService, ki je bil uporabljen za formiranje HTTP sporočil.

// slika ConfigureServices

// slika OpenReportServerCommunicationService

3.6.7 DashboardPage

// koga briga, spremeni

Na strani DashboardPage se uporabniku prikaže seznam odprtih projektov. Ko uporabnik klikne na željeni projekt, ga preusmeri na stran ProjectPage, kjer ga lahko ureja. Če nanj pridrži, se mu pokažejo dodatne opcije, kot so izbris in ahriviranje.

// slika DashboardPage

3.6.8 ProjectPage

Najpomembnejši del aplikacije je urejanje zapiskov. To lahko uporabnik dela,

3.7 Evalvacija funkcionalnosti

Poglavje 4

Možnosti nadaljnega razvoja

Poglavje 5

Zaključek